G'SCHEID SCHLAU!

DAS LANGE WOCHENENDE DER WISSENSCHAFTEN

Ostbayerische Technische Hochschule Amberg-Weiden

# Python programmieren mit dem 5€-Computer Raspberry Pi Pico – Ein Workshop mit der innovativen Mikrocontroller-Hardware

**Prof. Dr.-Ing. Ulrich Schäfer, Professor für Medieninformatik, Mobile Computing**

# Ostbayerische Technische Hochschule Amberg-Weiden

- Die OTH Amberg-Weiden ist eine junge, dynamische Hochschule und hat dir eine Menge zu bieten: kleine Seminargruppen, individuelle Betreuung, internationale Partnerhochschulen, renommierte Unternehmenspartner, und, und, und.
- Standorte in Amberg und Weiden
- Attraktive und bezahlbare Wohnungen sowie geringe Lebenshaltungskosten.
- 55 Studiengänge aus den Bereichen Technik, Wirtschaft, Informatik & Medien, Gesundheit, Energie & Umwelt sowie Pädagogik.
- 3800 Studierende

# Vorstellung

Prof. Dr. Ulrich Schäfer Homepage: https://www.oth-aw.de/schaefer-ulrich/

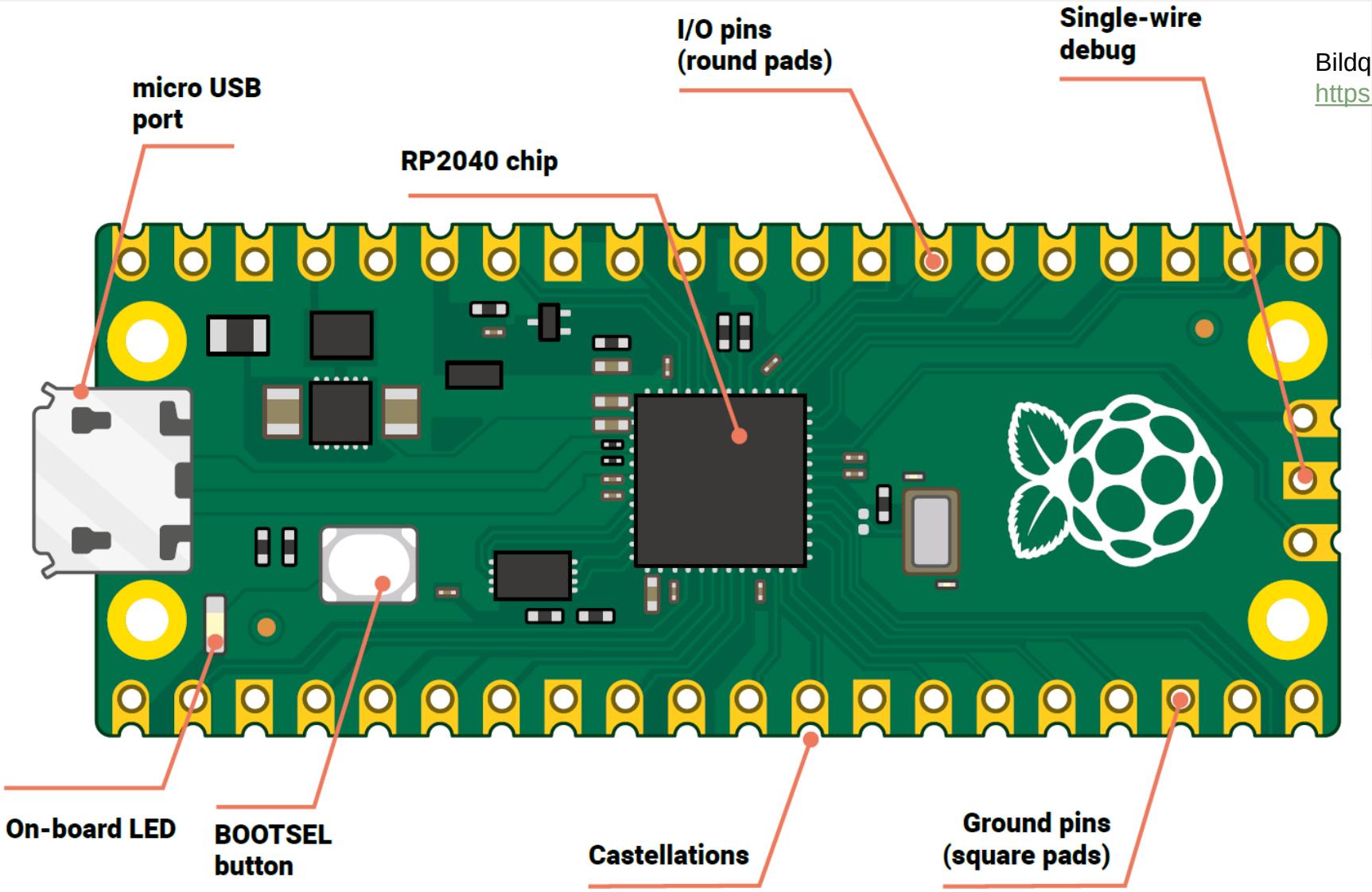
Ostbayerische Technische Hochschule Amberg-Weiden

Fakultät Elektrotechnik, Medien und Informatik (Standort Amberg)

Lehrgebiete: Medieninformatik & **Mobile Computing**

Forschung: + Natural Language Processing

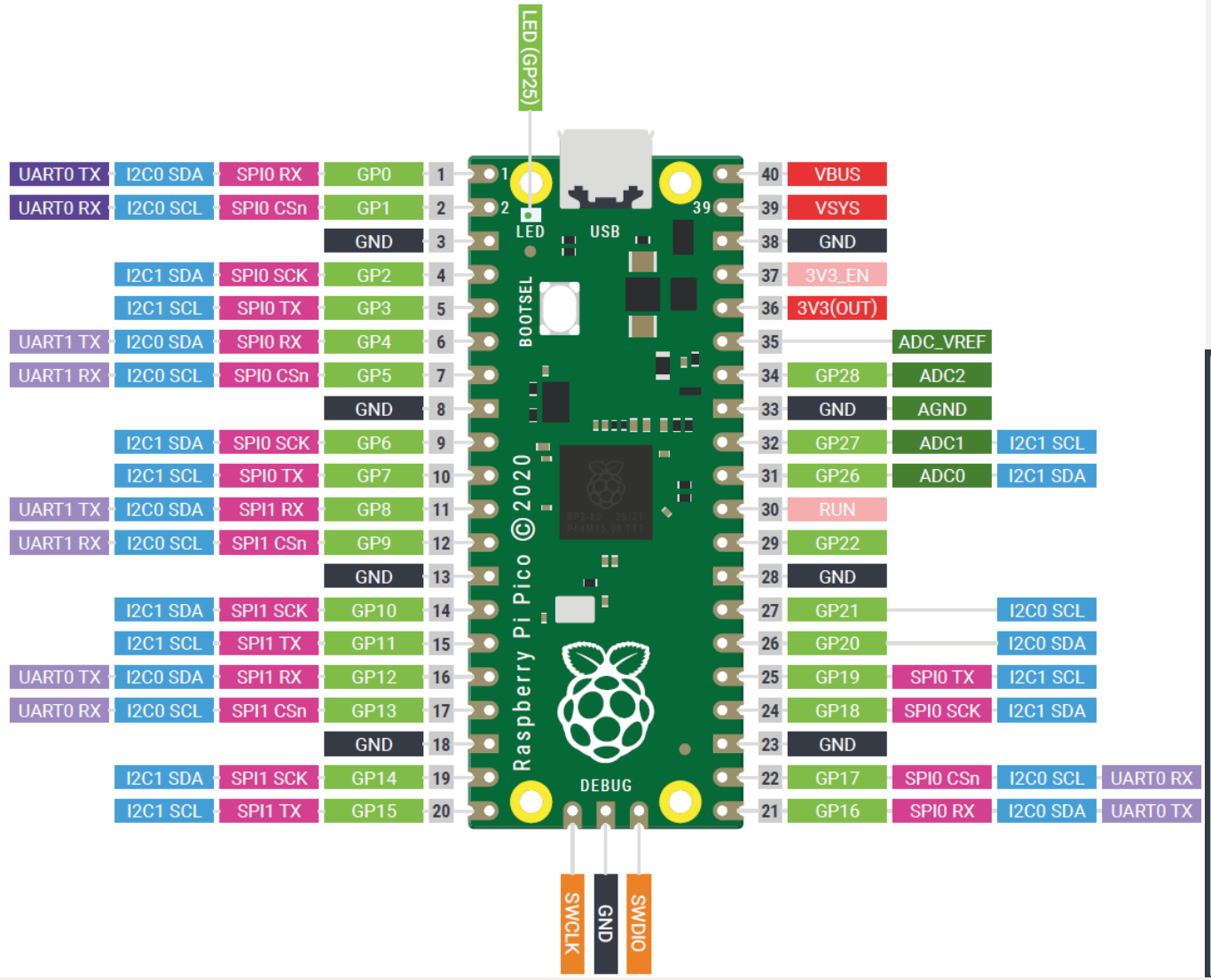
# Raspberry Pi Pico (RP2040)




Bildquellen:
https://www.raspberrypi.com/documentation/microcontrollers

- **CPU:** 32-bit dual-core ARM Cortex-M0+ at 48MHz, configurable up to 133MHz
- **RAM:** 264kB of SRAM in six independently configurable banks
- **Storage:** 2MB external flash RAM
- **GPIO:** 26 pins
- **ADC:** 3 × 12-bit ADC pins
- **PWM:** Eight slices, two outputs per slice for 16 total
- **Clock:** Accurate on-chip clock and timer with year, month, day, day-of-week, hour, second, and automatic leap-year calculation
- **Sensors:** On-chip temperature sensor connected to 12-bit ADC channel)
- **LEDs:** On-board user-addressable LED
- **Bus Connectivity:** 2 × UART, 2 × SPI, 2 × I2C, Programmable Input/Output (PIO)
- **Hardware Debug:** Single-Wire Debug (SWD)
- **Mount Options:** Through-hole and castellated pins (unpopulated) with 4 × mounting holes
- **Power:** 5 V via micro USB connector, 3.3 V via 3V3 pin, or 2–5V via VSYS pin

# Anschlussbelegung



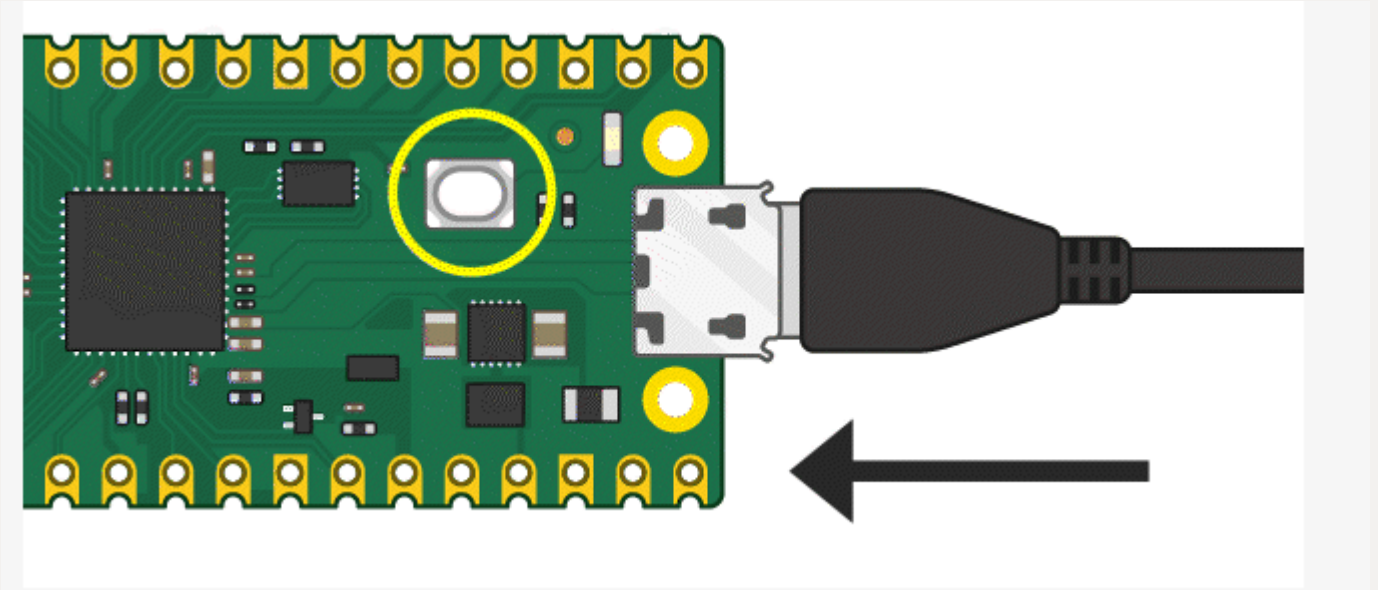

Bildquellen:
https://www.raspberrypi.com/documentation/microcontrollers

# MicroPython installieren

Ostbayerische Technische Hochschule
Amberg-Weiden

Bildquellen:
https://www.raspberrypi.com/documentation/microcontrollers

Anleitung (Animation):
https://www.raspberrypi.com/documentation/microcontrollers/micropython.html#drag-and-drop-micropython



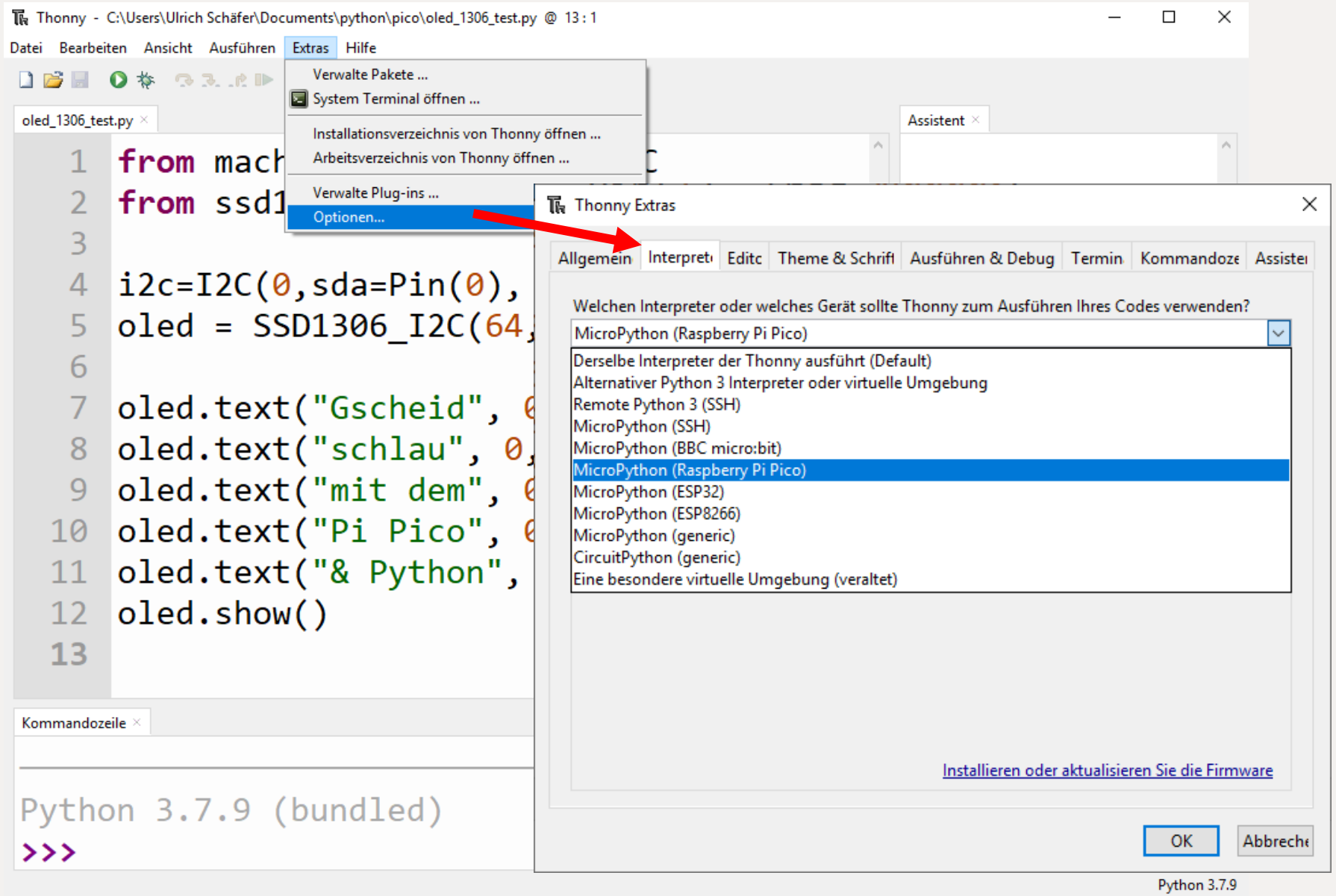
Direktlink zum MicroPython-Download: https://micropython.org/download/rp2-pico/rp2-pico-latest.uf2

# Entwicklungsumgebung: Thonny

Ostbayerische Technische Hochschule
Amberg-Weiden

- Download von https://thonny.org
- Einfach zu bedienende Python- Entwicklungsumgebung
- Code-Vervollständigung
- unterstützt interaktiven Python-Interpreter "remote" (auf dem per USB angeschlossenen Pico)
- Debugger und Variablen-Inspektor

# Entwicklungsumgebung: Thonny




Interpreter ändern auf MicroPython (Raspberry Pi Pico)

# MicroPython

Ostbayerische Technische Hochschule
Amberg-Weiden

- https://www.micropython.org
- "Abgespeckte" Version des "richtigen" Python 3 (CPython) von https://www.python.org
- https://www.raspberrypi.com/documentation/microcontrollers/micropython.html#what-is-micropython
- Wie beim "richtigen" Python: REPL (Read-Eval-Print-Loop) = interaktiver Interpreter
- Dokumentation zu MicroPython allgemein und Pico-spezifisch:
  - http://docs.micropython.org/en/latest/
  - http://docs.micropython.org/en/latest/rp2/quickref.html

# Hello World und Blink

Ostbayerische Technische Hochschule
Amberg-Weiden

```
import machine
leuchtdiode = machine.Pin(25, machine.Pin.OUT)

print("Hallo Welt")

print("Licht an")
leuchtdiode.value(1)

print("Licht aus")
leuchtdiode.value(0)
```

# Hello World und Endlos-Blinken

Ostbayerische Technische Hochschule
Amberg-Weiden

```
import machine, utime
leuchtdiode = machine.Pin(25, machine.Pin.OUT)

print("Hallo Welt")

while True:
  print("Licht an")
  leuchtdiode.value(1)
  utime.sleep(0.5)

  print("Licht aus")
  leuchtdiode.value(0)
  utime.sleep(0.5)
```

# Warum läuft mein Programm nicht nach dem Einschalten?

Ostbayerische Technische Hochschule
Amberg-Weiden

- (im Gegensatz zum Arduino): Interpreter im Flash-Speicher, Python-Programm im Hauptspeicher

- Lösung für dauerhaften Einsatz: Programm in **main.py** umbenennen und in das Dateisystem kopieren (BOOTSEL-Taste beim Anschließen drücken wie beim Installieren des Micro-Python-Interpreters auf dem Pico)

# Notrufsender (SOS)

Ostbayerische Technische Hochschule
Amberg-Weiden

```
import machine, utime
leuchtdiode = machine.Pin(25, machine.Pin.OUT)

def an_aus(zeit):
  leuchtdiode.value(1)
  utime.sleep(zeit)
  leuchtdiode.value(0)
  utime.sleep(zeit)

def S():
  for i in range(3):
    an_aus(0.2)
  utime.sleep(0.3)

def O():
  for i in range(3):
    an_aus(0.5)
  utime.sleep(0.3)

S(); O(); S();
```

# Thermometer

## Verwendung des im RP2040 eingebauten Temperatursensors

Ostbayerische Technische Hochschule
Amberg-Weiden

```
from machine import ADC

tempsensor = ADC(ADC.CORE_TEMP)

def Temperatur():
  messwert = tempsensor.read_u16() * 3.3 / 65535
  temp = 27 - (messwert - 0.706)/0.001721
  return temp

print("Temperatur: {:3.2f} °C".format(Temperatur()))
```

# Thermometer mit Datenlogger

## Platz für 1,3 MB Daten im Flash-Speicher

Ostbayerische Technische Hochschule
Amberg-Weiden

```
import utime

file = open("Temperaturen.txt", "w")

for i in range(86400 / 30): # Ca. einen Tag lang alle 30 Sekunden aufzeichnen
  file.write(str(Temperatur()))
  file.flush()
  utime.sleep(30) # 30 Sekunden warten

file.close()
```

# Datenlogger: Zeitstempel erzeugen

Ostbayerische Technische Hochschule
Amberg-Weiden

```
from machine import RTC
# format-Doku:  https://pyformat.info/
ZEITFORMAT = "{}-{:02d}-{:02d} {:02d}:{:02d}:{:02d} " # Jahr-Monat-Tag Stunde:Min:Sek

def Zeit():
  t = RTC().datetime()
  return ZEITFORMAT.format(t[0],t[1],t[2],t[4],t[5],t[6])

print(Zeit());

#Ausgabe:
2021-10-22 22:38:31
```

# Datenlogger: Zeitstempel in die Datei schreiben

Ostbayerische Technische Hochschule
Amberg-Weiden

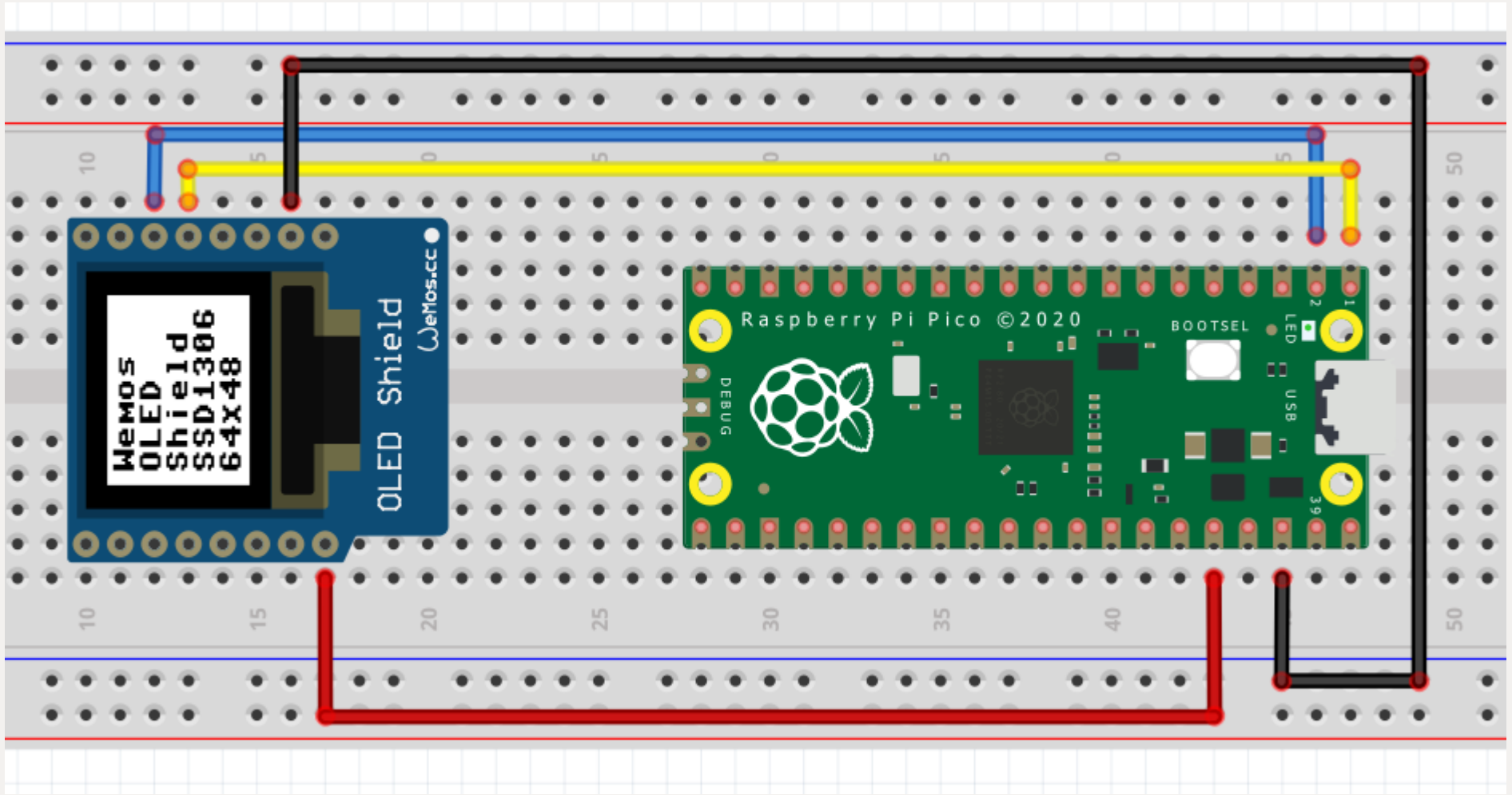
```
file = open("Temperaturen.txt", "w")

for i in range(86400 / 30): # einen Tag lang aufzeichnen (alle 30 Sekunden)
  file.write("{:s} {:3.2f}\n".format(Zeit(), Temperatur()))
  file.flush()
  utime.sleep(30) # 30 Sekunden warten

file.close()

file = open("Temperaturen.txt") # Daten lesen und ausgeben
print(file.read())
file.close()
```

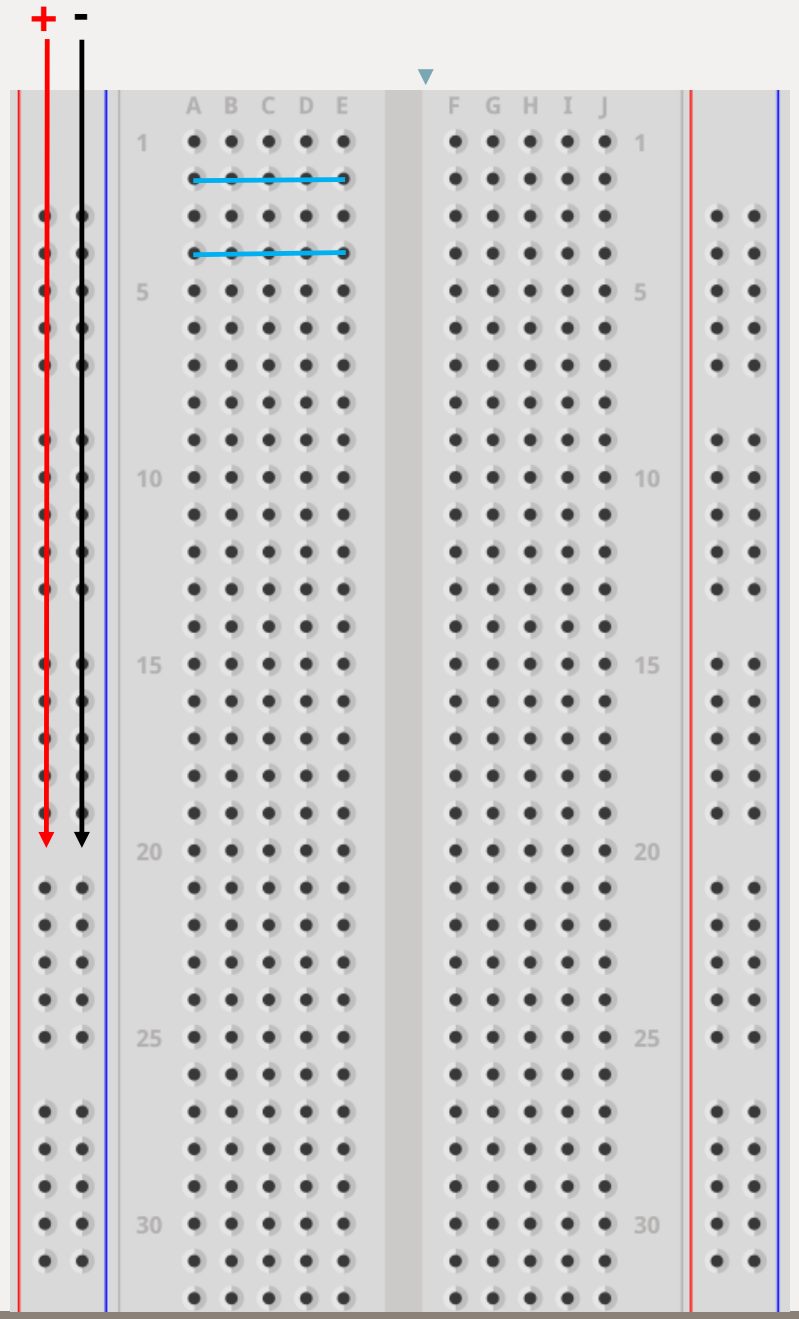
# OLED Shield anschließen




| | Pico | Display |
|---|---|---|
| schwarz | = GND | GND |
| rot | = 3.3V | 3.3V |
| gelb | = 1 | D2 |
| blau | = 2 | D1 |

Pico Fritzing-Modell und Pinout: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

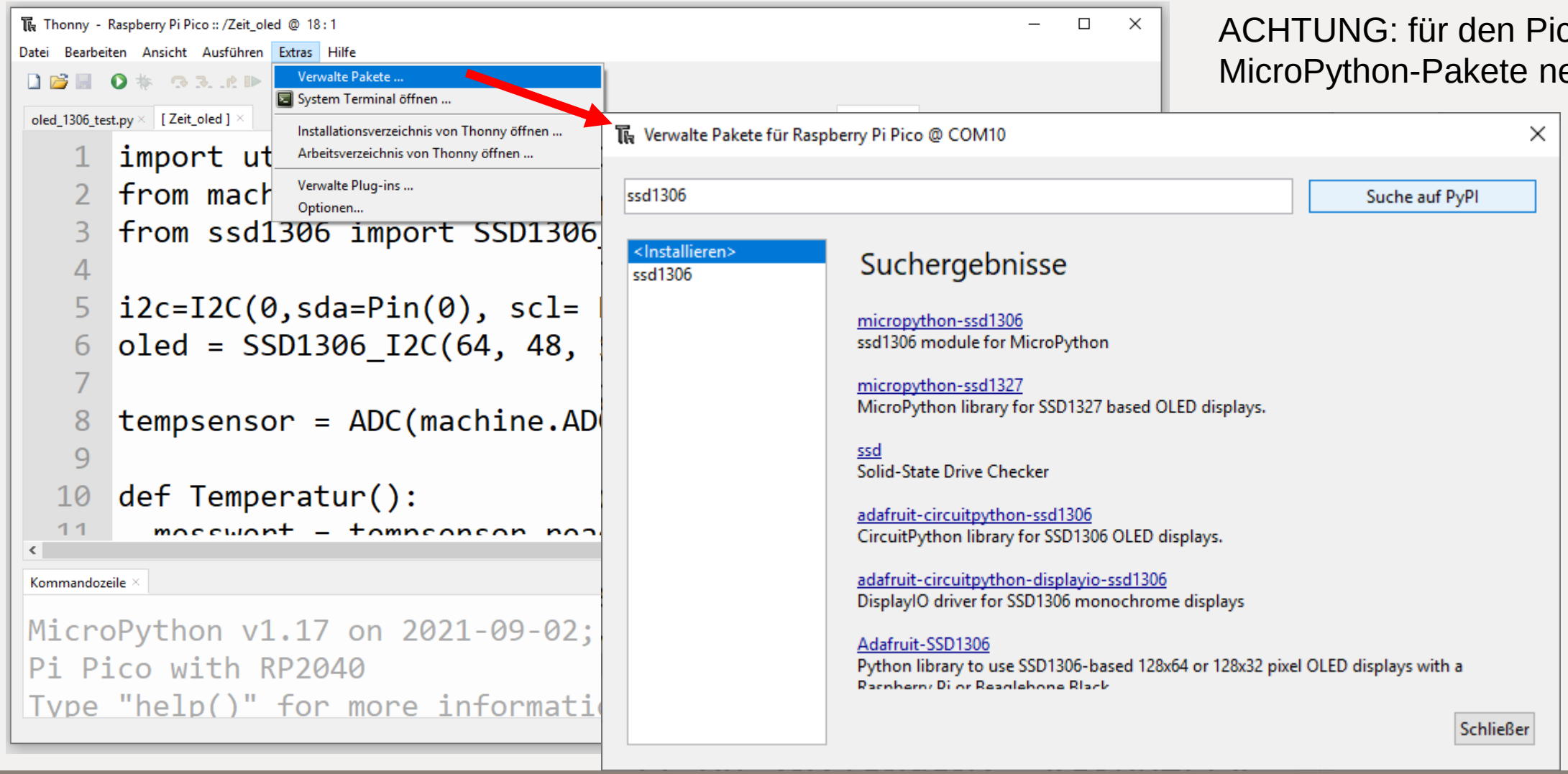
# Breadboard-Kontaktierung




**Achtung** - längs nicht immer durchgehend verbunden →

# Pakete für OLED-Display (SSD1306) und andere Spezialitäten nachinstallieren

Ostbayerische Technische Hochschule Amberg-Weiden




ACHTUNG: für den Pico nur MicroPython-Pakete nehmen!

# Uhr, Thermometer mit OLEDisplay (SSD1306)

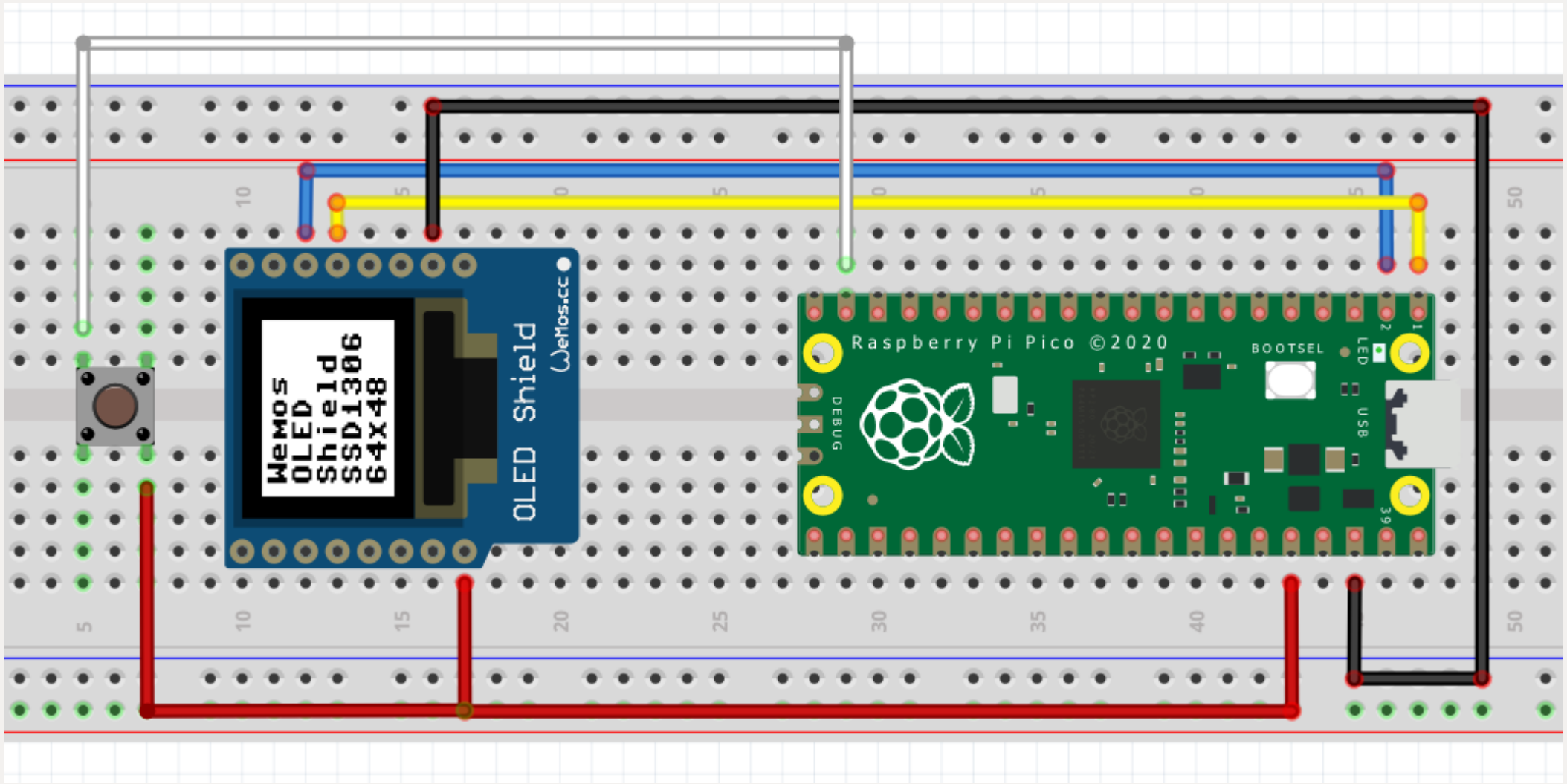
Ostbayerische Technische Hochschule
Amberg-Weiden

```
from machine import Pin, I2C, ADC, RTC
from ssd1306 import SSD1306_I2C   # ssd1306

i2c=I2C(0, sda=Pin(0), scl= Pin(1), freq=400000)
oled = SSD1306_I2C(64, 48, i2c)     # bei 64 x 48 Punkt-Display

while True:
  oled.fill(0);
  oled.text(Zeit()[2:11], 0, 0)
  oled.text(Zeit()[11:], 0, 20)
  oled.text("{:3.2f} ^C".format(Temperatur()), 0, 40)
  oled.show()
  utime.sleep(1)
```

# Taster anschließen (Pull-down)



# Taster-Tester

Ostbayerische Technische Hochschule
Amberg-Weiden

```
import machine
import utime
import _thread
taster = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_DOWN)
leuchtdiode = machine.Pin(25, machine.Pin.OUT)
global button_pressed
button_pressed = False

def button_reader_thread():
  global button_pressed
  while True:
    if taster.value() == 1:
      button_pressed = True
    utime.sleep(0.01)

_thread.start_new_thread(button_reader_thread, ())

while True:
  if button_pressed:
    leuchtdiode.value(1)
    utime.sleep(0.5)
    leuchtdiode.value(0)
    global button_pressed
    button_pressed = False
```

# Grafik auf dem OLEDisplay

Ostbayerische Technische Hochschule
Amberg-Weiden

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C  # ssd1306

i2c=I2C(0, sda=Pin(0), scl= Pin(1), freq=400000)
oled = SSD1306_I2C(64, 48, i2c)  # bei 64 x 48 Punkt-Display

oled.fill_rect(1, 1, 20, 20, 1)  # Rechteck zeichnen (x , y, Breite, Höhe, an/aus)
oled.show()
oled.pixel(1, 1, 1)              # einzelnen Punkt setzen (x , y, an/aus)
oled.show()
```

# Pong-Spiel: Ball wird von Wand reflektiert + Schläger, mit einem oder zwei Tastern gesteuert

Ostbayerische Technische Hochschule
Amberg-Weiden

- Idee: Ball-Bewegung in Endlosschleife
- Zeitmessung mit RTC
- Variablen: Position, Geschwindigkeit, Richtung, Punkte
- Reflektion am Display-Rand
- Schläger als Linie am unteren Bildschirmrand
- Ball als Rechteck oder Bitmuster (vgl. Himbeere im MicroPython-Buch am Ende)

# Grafik auf dem OLEDisplay

Ostbayerische Technische Hochschule
Amberg-Weiden

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C   # ssd1306
xmax = 64
ymax = 48 # bei 64 x 48 Punkt-Display
i2c=I2C(0, sda=Pin(0), scl= Pin(1), freq=400000)
oled = SSD1306_I2C(xmax, ymax, i2c)

def ball(x, y, malen=1): # 1=malen, 2=löschen
  oled.fill_rect(x, y, 2, 2, malen)
  oled.show()

def schlaeger(x, malen=1):
  oled.fill_rect(x, ymax - 2, 6, 2, malen)
  oled.show()

ball(20,20)
schlaeger(30) # Anregungen z.B. https://github.com/gurgleapps/Pico-Pong
```

# Ausblick

- Programmable I/O z.B. für schnelle Steuerung von LED-Streifen/Ringen, digitale Bildsignalerzeugung, Sound, ...
- CircuitPython mit USB HID-Emulation (Pico als USB-Eingabegerät)
- weitere Hardware: Lagesensor, Näherungssensor, Servomotor,...

Weitere Infos:

- https://www.raspberrypi.com/documentation/microcontrollers/
- https://www.oth-aw.de/schaefer-ulrich/pico/

**Danke fürs Mitmachen und weiterhin viel Spaß!**